FGV EMAp ESCOLA DE MATEMÁTICA APLICADA

Simulado A2 de Linguagens de Programação
17 de Novembro de 2025

Nome: _____

Questão 1 (4 pontos)

A plataforma **BookMatch** deseja facilitar a troca e doação de livros usados entre usuários. Cada **Usuário** pode anunciar livros, solicitar trocas e avaliar outros usuários após uma transação. Um **Livro** possui título, autor, gênero e estado de conservação (por exemplo: "novo", "bom", "gasto"). Uma **Transação** representa o processo de troca entre dois usuários e deve registrar o status (pendente, concluída ou cancelada) e a data da troca.

Implemente um modelo orientado a objetos em **Python** que represente esse cenário. Inclua:

1. As classes principais, seus atributos e métodos relevantes;
2. As relações entre classes (composição, associação, herança, etc.);
3. Um pequeno trecho de código de uso, simulando uma troca entre dois usuários.

Questão 2 (6 pontos)

Considere o trecho de um sistema legado utilizado para gerar, exportar e enviar relatórios de vendas e estoque.

```
class BaseReport:
    def __init__(self, data):
        self.data = data
        self.last_export = None

    def generate(self):
        print("Gerando relatório base. A empresa... (texto do relatório)")
        return str(self.data)

class SalesReport(BaseReport):
    def generate(self, detailed=False):
        print("Gerando relatório de vendas. A empresa... (texto do relatório)")
        content = f"Vendas: {self.data}"
        if detailed:
            content += " (detalhado)"
        print(content)
        return content

class InventoryReport(BaseReport):
    def generate(self):
```

```
        print("Gerando relatório de estoque. A empresa... (texto do relatório)")
        return f"Estoque: {self.data}"

class ReportService:
    def __init__(self, report):
        self.report = report

    def export(self, fmt="pdf"):
        content = self.report.generate()
        if fmt == "pdf":
            blob = ("[PDF]" + content).encode("utf-8")
        elif fmt == "csv":
            blob = ("[CSV]" + content).encode("utf-8")
        else:
            raise ValueError("Formato nao suportado")
        self.report.last_export = blob
        return blob

    def send(self, channel, target):
        if self.report.last_export is None:
            self.export("pdf")
        payload = self.report.last_export
        if channel == "email":
            print(f"Enviando e-mail para {target}: {payload[:30]}...")
        elif channel == "slack":
            print(f"Postando no Slack #{target}: {payload[:30]}...")
        else:
            raise ValueError("Canal nao suportado")

class Communicator:
    def send_email(self, payload, address):
        raise NotImplementedError

    def send_slack(self, payload, channel):
        raise NotImplementedError

    def send_sms(self, payload, phone):
        raise NotImplementedError

def main():
    sales = SalesReport(data=[10, 20, 30])
    service = ReportService(sales)
    blob = service.export(fmt="pdf")
    service.send(channel="email", target="financeiro@empresa.com")

    inv = InventoryReport(data={"SKU1": 5, "SKU2": 2})
```

```
    service2 = ReportService(inv)
    service2.export(fmt="csv")
    service2.send(channel="slack", target="relatorios")

if __name__ == "__main__":
    main()
```

Identifique e explique problemas de design no código legado, relacionando-os explicitamente a SOLI (sem o D). Refatore a hierarquia de relatórios e demonstre o uso do design refatorado.